REMARKS

Claim 1, as amended, calls for running at least two applications and enabling those applications to share a class. Further, the claim calls for duplicating the member data for the class for each application in a shared memory and then providing a handle to each application to enable the application to access its member data into shared memory.

The office action cites the reference to Czajkowski on a single reference 103 rejection. Such a rejection is almost certainly not statutory. If the reference fails to teach all of the elements, it cannot itself teach the rationale to modify itself. Rather than combine with other references, the Examiner has chosen to attempt to bolster the rejection by citing well known art. But there is no suggestion that the well known art provides any rationale to modify the cited reference.

In particular, official notice was taken that it is well known in the art that shared memory is a type of memory. The conclusion ("therefore, it would be obvious in view of Czajkowski in order to allow the applications to execute in shared memory and therein duplicate member data for a class so that the member data is stored in shared memory and accessed by applications by upon an identifier to the member data for the respected application") that is drawn from the official notice is not supported by that official notice. The fact that applications use shared memory does not teach a reason to use the shared memory to enable applications to share class data and, particularly, that this could be done by providing handles to those applications to enable them to access the shared data.

Further, it is asserted that it is well known in the art that when a client instantiates another class (an access method class), a reference to that object is returned to the requesting object. It is respectfully requested that the Examiner cite a reference in support of this assertion. Certainly, such a finding is obscure and it seems doubtful that this is well known. In particular, the Examiner is specifically called upon to show that it is well known in the art that when a client instantiates a methods class, a reference to that object is returned to the requesting object. It is not believed that there is any basis for this assertion.

The conclusion is drawn from this assertion of well known art that "the applications would receive a reference to the methods class when it is instantiated for accessing the application's respected number data." Of course, this begs the technical question. Once the

class is instantiated, there is no way for the applications to know about the class or to access it. Thus, not only is it technically unfeasible to suggest that what is claimed in the present application happens automatically, but it is certainly questionable that it does. In any case, if the Examiner can find such a reference, that will resolve the issue. If he cannot, and it is believed that he cannot, this is still another reason why the obviousness rejection fails.

In sum, a single reference obviousness rejection is substantiated, according to the office action, based on two different statements of well known art. Nowhere is there any showing of any rationale from within the art to make the claimed combination of the two well known asserted practices with the cited reference. Therefore, a *prima facie* rejection is not made out and reconsideration is respectfully requested.

Respectfully submitted,

Date: January 25, 2005

Timothy N. Trop, Reg. No. 28,994
TROP, PRUNER & HU, P.C.
8554 Katy Freeway, Ste. 100
Houston, TX 77024
713/468-8880 [Phone]
713/468-8883 [Fax]